**UNIVERSIDAD AUTONOMA DE MADRID**

**ESCUELA POLITECNICA SUPERIOR**




**Grado en Ingeniería Informática**

# TRABAJO FIN DE GRADO

## Modelo macroevolutivo para estudiar la evolución de la biodiversidad y la extinción de especies


**Juan Vega Najarro**
**Tutor: Roberto Latorre Camino**


**JUNIO 2019**

# Modelo macroevolutivo para estudiar la evolución de la biodiversidad y la extinción de especies

**AUTOR: Juan Vega Najarro**
**TUTOR: Roberto Latorre Camino**

**Dpto. Ingeniería Informática**
**Escuela Politécnica Superior**
**Universidad Autónoma de Madrid**
**Junio de 2019**

# Resumen

A lo largo de la historia, en nuestro planeta, han tenido lugar 5 grandes extinciones masivas. Dichas extinciones, siempre se han explicado como una consecuencia que surge a raíz de un cambio en el planeta producido por factor externo (como, por ejemplo: la caída de un meteorito). Pero en muy pocas ocasiones se han planteado como una consecuencia intrínseca de la propia dinámica sistema.

Este trabajo de fin de grado tiene como objetivo la implementación de un modelo capaz de simular las interacciones que ocurren entre las distintas poblaciones que forman una especie y, a su vez, con el entorno en el que estas habitan.

Para llevar a cabo dicha implementación, se ha seguido un riguroso orden: En primer lugar, se han buscado referencias biológicas para tener una base sólida de la que partir y en la que apoyarnos. Esto se debe a que en este proyecto tanto la biología como la ecología juegan un rol fundamental, al dictar cuáles son las reglas que debe seguir nuestro modelo, para que se adecue a la realidad. A continuación, se han investigado otros modelos que perseguían objetivos similares, con la finalidad de no inventar algo que ya ha sido creado y obtener sus principales ideas. Por último, se han analizado las tecnologías que mejor se adaptaban a los requisitos que exigía nuestro modelo, siendo uno de los factores fundamentales la eficiencia del código, ya que estamos hablando de simulaciones muy largas, con millones de especies en millones de épocas.

En cuanto a los resultados obtenidos con este modelo, cabe destacar la importancia que tienen los cambios bruscos en el hábitat de las especies, como podría ser el caso del calentamiento global, ya que son los principales responsables de las extinciones masivas. No obstante, este modelo sugiere de forma teórica que las extinciones masivas pueden ser un componente de la propia dinámica del sistema.

# Palabras clave

Macroevolución, modelo, extinción, evolución, especies, vida, simulación, Python, ecología, biología

# Abstract

Throughout history, on our planet, there have been 5 massive extinctions. These extinctions have always been explained as consequence that arises as a result of a change in the planet produced by an external factor (for example, the fall of a meteorite). But in very few occasions they have been considered as an intrinsic consequence of the dynamic system itself.

This Bachelor Thesis aims to implement a model capable of simulating the interactions that occur between the different populations that form a specie and, the environment where they live.

To carry out this implementation, a strict order has been followed: First, the biological references have been sought in order to have a solid base to start and support our advances. This is because in this project, biology and ecology play a fundamental role, by dictating what are the rules that our model must follow, so that it adapts to reality. Also, we have investigated other models that pursued similar objectives, in order that don’t invent something that has already been created, obtaining their main ideas. Finally, we have analysed the technologies that best fit with the requirements demanded by our model, one of the fundamental factors is the efficiency of the code, that’s because we are talking about very long simulations, with millions of species in millions of epochs.

Regarding the results obtained with this model, it is important to highlight the importance of sudden changes in the habitat of the species, such as the case of global warming. However, this model suggests theoretically that, mass extinctions can be a part of the system's dynamics.



# Keywords

Macroevolution, model, extinction, evolution, species, life, simulation, Python, ecology, biology

## *Agradecimientos*

*Quiero agradecer por todo su tiempo, entendimiento, consejos y rapidez a mi tutor del trabajo de fin de grado, Roberto Latorre ya que me ha guiado de forma excepcional. Ha sido todo un placer realizar el trabajo con él y volvería a hacerlo sin lugar a dudas.*

*También quiero agradecer a todos los profesores que me han dado clases o me han ayudado a lo largo de estos cuatro años de carrera ya que sin los conocimientos que me han enseñado no hubiese podido afrontar este proyecto.*

*Finalmente quiero agradecer a mis familiares y amigos que me han prestado su apoyo ya sea escuchándome o aportándome nuevas ideas ya que sin vuestra ayuda no podría haberlo conseguido.*

*Juan Vega Najarro.*

# INDICE DE CONTENIDOS

# INDICE DE FIGURAS

# 1 Introducción y Objetivos

## *1.1 Introducción*

En este trabajo vamos a diseñar e implementar un modelo macroevolutivo, con el fin de estudiar las dinámicas de aparición y extinción de especies, observadas en el registro fósil. Para ver si estas dinámicas son o no un efecto del propio sistema o se deben a factores externos.

La macroevolución, consiste en el análisis de los cambios evolutivos a nivel de especie. Mientras que un modelo es la herramienta que utilizamos para, a partir de un conjunto de especies y una serie de reglas, simular su evolución en un periodo de tiempo dado.

Una especie, es un conjunto de individuos que comparten una serie de características y, que además pueden reproducirse entre sí, originando una descendencia fértil. Dichas características, son representadas a través del ADN, el cual contiene la información genética de un individuo, o genoma.

Una regla, es la implementación lógica, o informática que le damos a un principio o ley de la naturaleza. Estas las extraeremos de diferentes fuentes, que veremos a lo largo de este documento.

## *1.2 Motivación*

Tanto en la escuela, como en la mayoría de las empresas tendemos a aplicar la informática para resolver problemas matemáticos, hacer aplicaciones que hagan nuestro día a día más fácil… Pero en muy pocas ocasiones orientamos la informática a la biología.

Para este trabajo de fin de grado, quería aprovechar la oportunidad y aprender como enfocar la informática para la realización de un modelo biológico.

Por todo esto, decidí embarcarme en este proyecto, el cual me daría la oportunidad de conocer más de cerca el mundo de la investigación. Además, de tener ese enfoque biológico, que me resulta atractivo.

## 1.3 Objetivos

Los objetivos que he perseguido en este proyecto, ordenados por el orden en el que los he ido cumpliendo, han sido:

- Repasar cuáles han sido las grandes extinciones de la historia y cuáles son sus causas más probables.
- Leer sobre ecología, para poder entender cómo se comportan las poblaciones de especies, tratando de identificar cuáles han sido los mecanismos fundamentales que han llevado a un aumento de la biodiversidad en nuestro planeta y así definir una serie de reglas para aplicar a mi modelo.
- Buscar y analizar artículos científicos sobre modelos o trabajos que buscaban un objetivo similar al mío, buscando nuevas ideas, sobre cómo otras personas abordaron un problema afín.
- Conocer cuáles son las herramientas más utilizadas en el mundo de la simulación y escoger una, en este caso fue Python.
- Diseñar cuál sería la estructura del modelo buscando siempre la forma más eficiente, ya que esto repercutirá en la velocidad del propio modelo.
- Estudiar técnicas de procesado de datos para aumentar la velocidad y rendimiento del modelo en simulaciones largas, buscando siempre el menor consumo posible de recursos.
- Implementar el modelo y adaptar sus parámetros para obtener resultados realistas.
- Analizar los resultados obtenidos.

## 1.4 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del arte:** Veremos cuáles son los datos más importantes hasta el momento, de cara a la implementación del modelo.
- **Diseño:** Analizaremos que es lo que queremos desarrollar y lo estructuraremos utilizando un análisis de requisitos, escogiendo las herramientas con las que implementar el proyecto y elaborando su diagrama de clases UML.
- **Desarrollo:** Explicaremos la lógica del modelo, así como las decisiones que fueron tomadas a la hora de su implementación.
- **Integración, pruebas, resultados:** Estudiaremos cuáles han sido los resultados del modelo para las pruebas realizadas y también, cuáles fueron los detalles más relevantes a la hora de poner el modelo en marcha.
- **Conclusiones y trabajo futuro:** Discutiremos cuáles podrían ser futuras mejoras para el modelo y cuál es la conclusión del proyecto en sí.

# 2 Estado del arte

## *2.1 Grandes extinciones a lo largo de la historia de la Tierra*

Al tratarse de un modelo macroevolutivo, las grandes extinciones son uno de los ejes fundamentales que determinan la evolución de las especies por ello, veremos cuáles han sido las cinco grandes extinciones que ha sufrido nuestro planeta y cuáles fueron sus causas más probables. Hemos de dejar claro que, aunque se asume que es una certeza, no es seguro que fuese como cuentan.

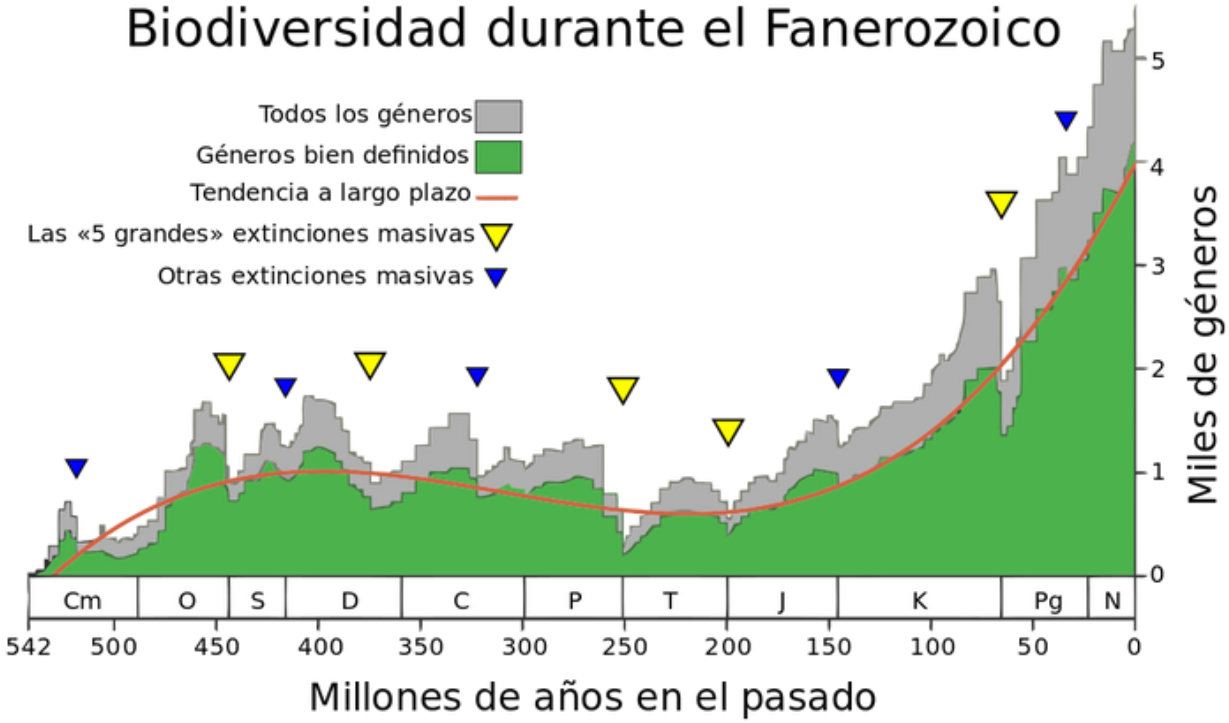


**Figura 2.1: Estimación de la evolución del número de géneros en la Tierra desde la explosión cámbrica hasta nuestros días. Figura adaptada de [2].**

La Figura 2.1 muestra una estimación de cómo ha sido la evolución del número de géneros de especies desde la explosión cámbrica [2] hasta la actualidad. La tendencia general es un aumento en el número de especies a medida que pasa el tiempo. Sin embargo, existen periodos en los que esta tendencia general se rompe y el número de especies disminuye. En algunos casos, la disminución es pequeña. En otros, conocidos como *grandes extinciones masivas*, la disminución es muy significativa. Estos eventos de extinción son uno de los ejes fundamentales en la dinámica macroevolutiva de las especies. Apoyándonos en la Figura 1, vamos a repasar cuáles han sido las cinco (¿seis?) grandes extinciones que ha sufrido nuestro planeta, así como cuáles son las causas mayormente aceptadas por la comunidad científica para cada una de ellas.

Además, dicha figura también es el objetivo final de nuestra simulación, ya que, si logramos obtener una gráfica similar, significará que las grandes extinciones, podrían considerarse un proceso intrínseco de la dinámica del sistema.

| Eón | Era | Período | Época | Millones años |
|---|---|---|---|---|
| Fanerozoico | Cenozoico | Cuaternario | Holoceno | 0,011784 |
| | | | Pleistoceno | 2,588 |
| | | Neógeno | Plioceno | 5,332 |
| | | | Mioceno | 23,03 |
| | | Paleógeno | Oligoceno | 33,9 ±0,1 |
| | | | Eoceno | 55,8 ±0,2 |
| | | | Paleoceno | 65,5 ±0,3 |
| | Mesozoico | Cretácico | | 145,5 ±4,0 |
| | | Jurásico | | 199,6 ±0,6 |
| | | Triásico | | 251,0 ±0,4 |
| | Paleozoico | Pérmico | | 299,0 ±0,8 |
| | | Carbonífero | Pensilvánico | 318,1 ±1,3 |
| | | | Misisípico | 359,2 ±2,5 |
| | | Devónico | | 416.0 ±2,8 |
| | | Silúrico | | 443,7 ±1,5 |
| | | Ordovícico | | 488,3 ±1,7 |
| | | Cámbrico | | 542,0 ±1,0 |

**Figura 2.2: Gráfica división del Fanerozoico [2]**

En la figura 2.2, vemos con más detalle la duración y las eras del Fanerozoico. Hacemos referencia a estas escalas de tiempo debido a que son una unidad geocronológica, y en este modelo hablamos de macroevolución, por lo cual, es lógico pensar en estos términos. Si el lector tiene curiosidad acerca de los eones, en la referencia [3], encontrará más información al respecto. En este apartado no entraremos en más detalles ya que no aportan ninguna información imprescindible para el modelo.

La primera gran extinción es la “extinción masiva del Ordovícico-Silúrico", donde desaparecieron el 86% de las especies que habitaban nuestro planeta en aquel entonces. En estos periodos solo podíamos encontrar vida en el mar ya que en la superficie no abundaba el oxígeno. Sobre la causa, existen distintas corrientes, entre las que destacan: un periodo glaciar, una disminución drástica de la cantidad de oxígeno o la explosión de una supernova [1].

La siguiente gran extinción es "extinción masiva del Devónico-Carbonífero", con una desaparición del 82% de las especies vivas en el planeta. Respecto a la causa todos los estudios apuntan a que, tras la aparición de las plantas en la superficie terrestre, sus raíces, liberaron nutrientes al océano que produjeron un aumento de las poblaciones de algas, aumentando el consumo de oxígeno del agua, sentenciando la vida animal [1].

Ahora veremos la mayor extinción que ha sufrido nuestro planeta, "extinción masiva del Pérmico-Triásico", donde desaparecieron el 96% de las especies que habitaban la Tierra. Dicha extinción también se conoce como “Un apocalipsis volcánico” ya que se trató de un periodo de alta actividad volcánica, lo cual liberó una gran cantidad de gases de carbono, desencadenado un **calentamiento global** que produjo tal catástrofe [1].

La siguiente gran extinción en la lista "extinción masiva del Triásico-Jurásico", terminó con la vida del 76% de las especies. Se ha responsabilizado por esta gran extinción al cambio climático, producido a raíz de las erupciones volcánicas constantes provocadas por la fragmentación del supercontinente Pangea. También hay otra teoría que indica que la gran extinción pudo estar relacionada con el impacto de un asteroide [1].

Por último, con una tasa de desaparición del 76% de las especies, la más famosa de las extinciones, "extinción masiva del Cretácico-Paleógeno", en la cual se extinguieron los dinosaurios. El motivo más apoyado fue que un gran asteroide colisionó con la corteza terrestre creando el cráter de Chicxulub, en la Península de Yucatán [1].

Hablamos de la sexta gran extinción masiva, al referirnos al impacto que el ser humano está produciendo sobre la tierra, como, por ejemplo, con la liberación de un exceso de gases de efecto invernadero, destruyendo el hábitat de diversas especies…

En este apartado hemos repasado brevemente las principales causas aceptadas por la comunidad científica para la desaparición de especies en la historia de la Tierra. El factor común para todas las hipótesis mayoritariamente aceptadas para las grandes extinciones es un cambio en las características extrínseco del área donde estas especies habitaban.

## 2.2 Fuerzas macroevolutivas

El primer punto importante que debe ser tenido en cuenta es que las poblaciones de una misma especie no permanecen siempre en la misma área, sino que tienden a expandirse por las áreas cercanas (ver ejemplos en Figuras 2.3 y 2.4).

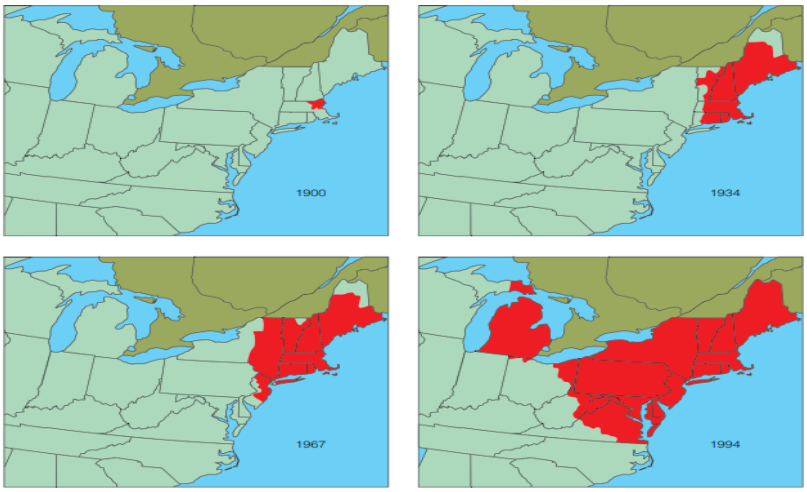


**Figura 2.3: Extensión de mariposas lagartas en América [4]**

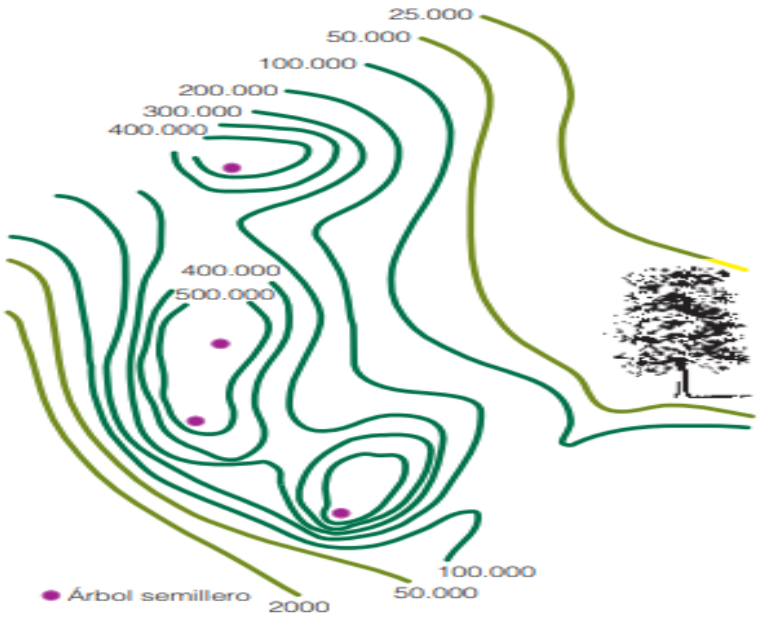


**Figura 2.4: Distribución de semillas del tulipero de Virgina [4]**

En muchas ocasiones hemos oído que la evolución favorece la "supervivencia del más fuerte". Sin embargo, esto es un claro error en la interpretación de las teorías de Malthus, Wallace y Darwin. Lo que realmente se favorece a nivel microevolutivo es la supervivencia (y reproducción) de los individuos mejor adaptados, o usando la terminología más especializada, aquellos con un mejor *fitting*. Esto puede ser; el más fuerte, el que mejor se camufla, el más oportunista o cualquier otra característica que haga que el individuo compita en su medio mejor que los demás. Entonces, los genes que aportan esa característica física, de comportamiento… van a verse favorecidos probabilísticamente en la siguiente generación.

En este apartado también debemos incluir cuál es el proceso de especiación; es decir, los mecanismos que conducen a la aparición de una nueva especie. Para ello, en primer lugar, hablaremos del anillo de especies que ilustra la figura 2.5, donde podemos ver que se ha producido un proceso de evolución al no poder reproducirse cuando se trata de cerrar el anillo, debido a que los genes de las distintas poblaciones han mutado a lo largo del tiempo.

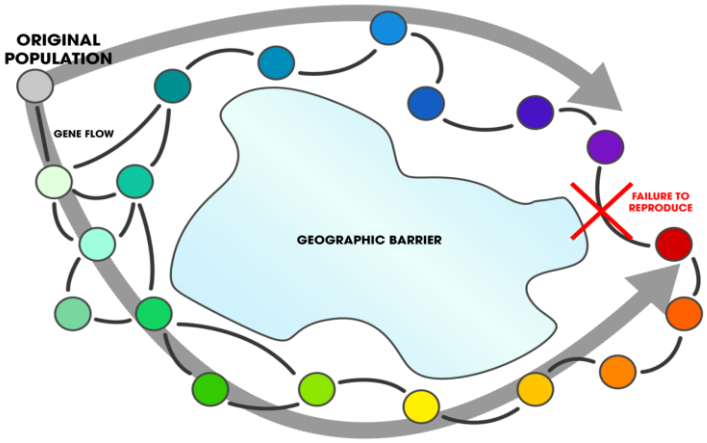


**Figura 2.5: Proceso de especiación [6]**

Jerry Coyne y H. Allen Orr señalan que las especies de anillos más de cerca modelan la especiación parapátrica [6]. Típicamente existen cuatro mecanismos de especiación (figura 2.6).

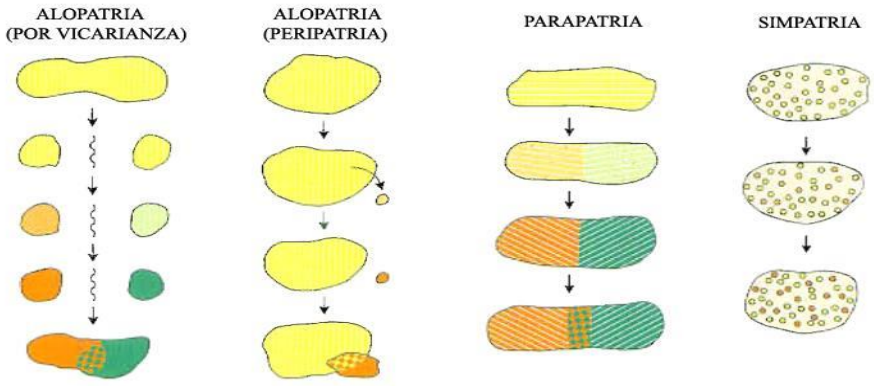


**Figura 2.6: Proceso de especiación de J.A. Coyne y H.A. Orr [7]**

En la especiación alopátrica, ya sea por vicarianza o peripátrica, hay barreras extrínsecas, por ejemplo, una montaña, que hacen que las poblaciones empiecen a divergir genéticamente. Sin embargo, en parapatría y simpatría hay barreras intrínsecas, y aquí hay bastante más controversia de que en realidad ocurra.

Para finalizar con este apartado debemos remarcar con otra definición de especiación: “es un proceso lento que va haciendo que las nuevas generaciones vayan siendo diferentes, hasta llegar al punto de que ya no podrían reproducirse con la población progenitora inicial de dicha especie dando lugar a una nueva” [8].

En resumen, tenemos cuatro fuerzas de suma importancia en el proceso evolutivo a nivel de población [9]:

- **Selección**: Los mejor adaptados son los que sobreviven y se reproducen.

- **Deriva genética**: Todos los individuos no se reproducen con la misma probabilidad, por ello se pierde variabilidad genética.

- **Mutación**: Variación del genoma de un individuo a lo largo del tiempo.

- **Migración**: Los individuos de distintas poblaciones tienden a moverse hacia un hábitat diferente, en general esto tiende a homogeneizar el genoma las poblaciones.

## *2.3 Darwinismo y Neodarwinismo*

Al hablar de un modelo que se basa en la evolución de las especies. No podíamos no mencionar a Charles Darwin, el hombre que escribió "el origen de las especies". En su obra habla de que todas las especies provienen de un antecesor común, además de plantear "la selección natural" [10]. Los principales factores que influyen en la selección natural son:

- **Variabilidad**: Los individuos presentan características diferentes producidas de manera aleatoria. Algunas favorecen la supervivencia y el éxito reproductivo del individuo, mientras que otras resultan perjudiciales. Si estas variaciones son heredables y no son producidas por el ambiente, serán relevantes en el proceso evolutivo.

- **Lucha por la existencia**: Darwin observó que el número de individuos que forman las poblaciones se mantiene constante, ya que de los individuos nacidos no todos logran alcanzar la madurez.

- **Reproducción diferencial**: Los individuos que cuentan con mejores características se reproducen con más éxito, por lo que las siguientes generaciones portarán sus rasgos. Los individuos menos aptos, no alcanzan la madurez o dejan menos descendientes, por lo que sus características tienden a desaparecer.

El neodarwinismo surge como una versión actualizada, basada en los avances científicos de la época, principalmente debido a los conocimientos en genética. Las principales modificaciones de la teoría fueron:

- Definen evolución como un cambio progresivo a lo largo del tiempo.

- Llamaremos mutaciones al cambio en las características heredables que aparecen de forma espontánea en una población, siendo esta la unidad evolutiva, en lugar del individuo como vimos en el darwinismo.

- La selección la produce tanto la reproducción diferencial como el impacto ambiental.

## *2.4 Modelos de dinámica de poblaciones*

Cuando se habla de modelos de dinámica de poblaciones típicamente se hace referencia a modelos de evolución que tratan de simular cómo cambiaría con el tiempo el tamaño de una población. Es decir, partiendo de un tamaño de población inicial dado, el modelo calcularía cuál sería el tamaño de la población después de un periodo de tiempo $\Delta t$ (un segundo, una hora, un día...). Y así sucesivamente hasta el final de la simulación.

El modelo de evolución más sencillo es el modelo de evolución lineal. En este modelo, por cada unidad de tiempo que pasa, la variable respuesta, es decir, el tamaño de la población varía una cantidad fija que puede ser una positiva o negativa. Obviamente, éste es un modelo teórico que no puede aplicarse a ninguna población biológica.

Un segundo modelo de evolución muy común en el crecimiento de poblaciones es el modelo exponencial. En este modelo, por cada unidad de tiempo que pasa, la variable respuesta aumenta o disminuye un porcentaje de su valor en el periodo inmediatamente anterior. Este modelo se adapta muy bien al crecimiento bacteriano en una situación de recursos infinitos. Sin embargo, presenta un serio problema para ser considerado en un caso general y es que la población crece indefinidamente, por lo que solo es realista para periodos cortos de tiempo.

Una variante del modelo de evolución exponencial que palia alguna de sus carencias es el modelo de exponencial con eliminación o aumento fijo. Según este modelo, la evolución de la población, en una generación aumenta o disminuye un porcentaje determinado con respecto a la generación anterior, pero, además, se permite la eliminación o aumento de una cantidad fija de individuos en cada generación. Este modelo aplica, por ejemplo, en casos como el de una población que crece un porcentaje cada año, pero que se ve afectada por la caza en un valor constante.

Otro de los problemas de las distintas variantes del modelo exponencial es que cuando los individuos de una población deben competir por los recursos disponibles para sobrevivir, cuando los recursos son limitados, o cuando el número de individuos es muy grande, básicamente, cuando interviene la selección natural, el crecimiento de la población no podrá ser proporcional al número de individuos de la generación anterior. En este escenario surge el modelo de evolución logístico discreto, que impone un tamaño máximo a la población alrededor del cuál va a fluctuar.

Los modelos descritos hasta el momento calculan la dinámica de una población, pero en el mundo real las poblaciones no están aisladas, sino que interactúan unas con otras. Por ejemplo, en un ecosistema con especies.

Cuando hablamos de modelos de dinámica de poblaciones típicamente, nos referimos a modelos que representan las dinámicas de competición entre dos o más especies. En este contexto, uno de los principales modelos que debemos conocer es el modelo de Lotka-Volterra [4][11].

El modelo de Lotka-Volterra, nos ofrece un enfoque de la interacción entre presa y predador, mediante dos ecuaciones diferenciales (figura 2.7), cuya representación podemos ver en la figura 2.8. También, podemos encontrar un tutorial de como implementar en Python este modelo [13].

$$\frac{dx}{dt} = \alpha x - \beta xy,$$
$$\frac{dy}{dt} = \delta xy - \gamma y,$$

**Figura 2.7: Ecuaciones diferenciales Lotka-Volterra [11]**

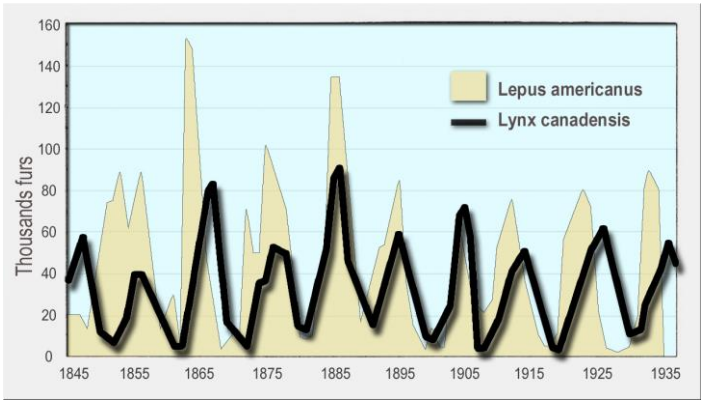


**Figura 2.8: Ejemplo ecuaciones con linces y conejos [11]**

Para nuestro estudio, el de Lotka-Volterra tiene un problema y es que reproduce la dinámica de competición solo, entre dos poblaciones interrelacionadas, mientras que nosotros necesitamos es simular la competición entre múltiples poblaciones que, además, aparecen y desaparecen en el tiempo.

El modelo de Lotka-Volterra generalizado, es la solución al problema planteado en el apartado anterior, el cual cuenta con un conjunto de ecuaciones (figura 2.9) que nos permite modelizar la competencia directa y las relaciones tróficas entre un número arbitrario de especies. Sin embargo, estas también cuentan con un problema y es que no tienen en cuenta algunos factores, como, por ejemplo; la preferencia de los depredadores… [12].

Las ecuaciones de Lotka-Volterra generalizadas modelan la dinámica de las poblaciones. $x_1, x_2, \ldots$ de $n$especies biológicas. Juntas, estas poblaciones pueden ser consideradas como un vector. $\mathbf{x}$. Son un conjunto de ecuaciones diferenciales ordinarias dadas por

$$\frac{dx_i}{dt} = x_i f_i(\mathbf{x}),$$

donde el vector $\mathbf{f}$ es dado por

$$\mathbf{f} = \mathbf{r} + A\mathbf{x},$$

dónde $\mathbf{r}$es un vector y A es una matriz conocida como la matriz comunitaria .

**Figura 2.9: Ecuaciones Lotka generalizadas [12]**

## 2.5 Evolución gramatical

Una herramienta que debemos mencionar es la evolución gramatical (GE), que, tras definir una serie de reglas (figura 2.9 y figura 2.10), y ofrecerle un dataset de entrada, de forma similar al comportamiento de un compilador, va evolucionando los datos de entrada al ejecutar una u otra regla. La figura 2.10 pertenece a un artículo donde nos ofrecen un ejemplo de evolución gramatical aplicado a las ecuaciones Lotka-Volterra, mencionadas en apartados anteriores.

$E ::= 0 \mid o0 \mid 0o0$ (mathematical expression)
$0 ::= N \mid X \mid (E)$ (operands)
$o ::= + \mid - \mid \times \mid * \mid \div \mid \bigcirc \mid \Gamma \mid \mathrm{L} \mid \otimes \mid ! \mid \mid$ (operators)
$N ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$ (digits)

**Figura 2.10: Conjunto de reglas GE [14]**

(1)⟨expr⟩ ::= ⟨expr⟩⟨op⟩⟨expr⟩ ..... rule 0
| (⟨expr⟩⟨op⟩⟨expr⟩) ..... rule 1
| ⟨pre-op⟩(⟨expr⟩) ..... rule 2
| ⟨var⟩ ..... rule 3
(2)⟨op⟩ ::= + ..... rule 0
| − ..... rule 1
| / ..... rule 2
| * ..... rule 3
(3)⟨pre-op⟩ ::= sin ..... rule 0
| Cos ..... rule 1
| Log ..... rule 2
(4)⟨var⟩ ::= X ..... rule 0
| 1.0 ..... rule 1

**Figura 2.11: Conjunto de reglas GEGA [15]**

A su vez, la figura 2.11 pertenece a otro artículo, donde explican una gramática evolucional mejorada, mediante el uso de un algoritmo genético. Es interesante, pero no entraremos en detalle ya que no es algo fundamental para este proyecto.

## *2.6 Enfoque de crecimiento en red en macroevolución*

En este punto hablaremos sobre un artículo en el cual diseñan un modelo con objetivos similares a los que nosotros perseguimos. Utilizando una red de relaciones y competencia entre especies logran unos resultados similares a los del registro fósil [16].

Las ideas principales de dicho modelo son:

- Crear una red de especies las cuáles tienen un peso que indica la intensidad que ejercen sobre la otra especie conectada.

- Las especies se distribuyen en un espacio bidimensional.

- La probabilidad de extinción de una especie se define en base al peso de las aristas conectadas a dicha especie.

- Las especies nacen a lo largo del tiempo en base a una tasa de especiación que se calcula en función del número de especies existentes y una constante que ellos definen. Limitando a su vez el número máximo de especies que puede habitar su mundo para aumentar la dificultad de aparición de especies progresivamente.

- Una vez ha nacido una nueva especie la añaden a la red y calculan sus relaciones en base a otra constante de radio.

- De forma aleatoria, por cada interacción deciden en base a la tasa de especiación si dicha especie debe dar lugar a una nueva especie o si la especie debe quedarse fuera de la red durante un intervalo de tiempo.

El resto del artículo es interesante ya que discuten los resultados de las simulaciones, que aparentemente son lógicos, los cuáles no comentaremos ya que a nosotros lo que nos interesa es como construyeron su modelo, con los cuáles obtuvieron esos resultados.

## *2.7 Juego de la vida*

El juego de la vida (figura 2.11), como vemos en [25]: es un juego de cero jugadores, lo que quiere decir que su evolución está determinada por el estado inicial y no necesita ninguna entrada de datos posterior. El tablero de juego es una malla plana formada por cuadrados (las células) que se extiende por el infinito en todas las direcciones. Por tanto, cada célula tiene 8 células vecinas. Las células tienen dos estados: están vivas o muertas. El estado de las células evoluciona a lo largo de unidades de tiempo. El estado de todas las células se tiene en cuenta para calcular el estado de estas al turno siguiente. Todas las células se actualizan simultáneamente en cada turno, siguiendo estas reglas:

- Una célula muerta con exactamente 3 células vecinas vivas "nace" (es decir, al turno siguiente estará viva).
- Una célula viva con 2 o 3 células vecinas vivas sigue viva, en otro caso muere (por "soledad" o "superpoblación").

Dicho modelo, a pesar de ser simple y poder resultar inútil de cara a lo que queremos implementar en este proyecto, no es así. Ya que nosotros, en lugar de células tendremos poblaciones y en vez de esas dos reglas simples tendremos reglas acordes al origen de poblaciones con la información obtenida con todo lo visto hasta ahora.

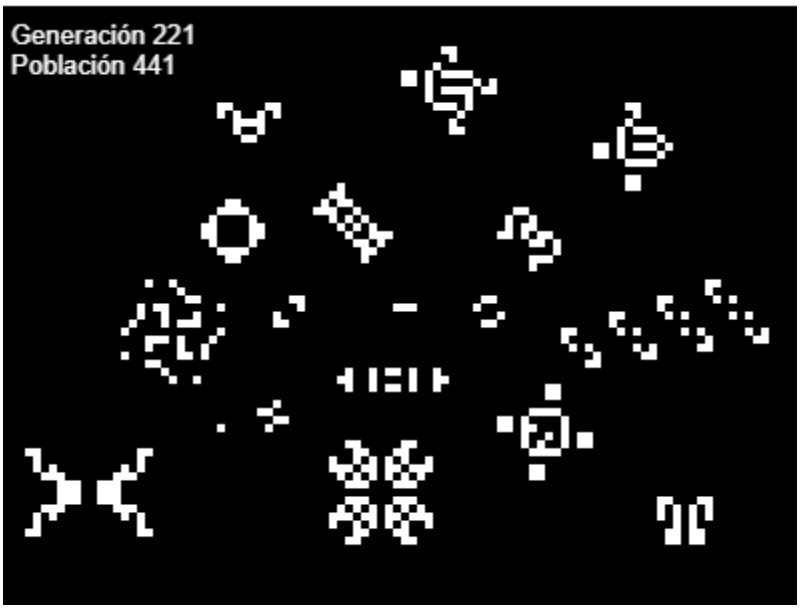


**Figura 2.12: Juego de la vida [25]**

# 3 Diseño

## *3.1 Análisis de requisitos*

Una vez terminada la investigación, nos faltaba ver cómo íbamos a plantear el problema. Para ello, el primer paso fue sintetizar toda la información, en una lista de requisitos funcionales y no funcionales que debía satisfacer el modelo.

### 3.1.1 Requisitos funcionales

Req(1): El modelo permitirá crear un planeta.
Req(2): El planeta tendrá unas dimensiones.
Req(3): El modelo permitirá crear y añadir áreas a un planeta.
Req(4): Las áreas tendrán unas características propias (clima, porcentaje de agua…).
Req(5): Las características de las áreas podrán verse alteradas por factores externos.
Req(6): El modelo permitirá tener especies y que estas vivan en determinadas áreas del planeta.
Req(7): Las especies estarán compuestas por poblaciones de individuos.
Req(8): Las especies podrán migrar a las áreas cercanas del planeta.
Req(9): Las poblaciones tendrán un genoma propio, que haga referencia a su fenotipo y del cuál dependerá su capacidad de adaptarse a una zona.
Req(10): Las poblaciones que no logren adaptarse a una determinada zona morirán.
Req(11): Las poblaciones podrán reproducirse entre ellas y con otras poblaciones de la misma especie dando lugar a nuevas poblaciones y especies.
Req(12): Las poblaciones tendrán un fitting.
Req(13): El genoma de una población puede mutar tanto por procesos reproductivos como por influencia de factores externos (ej. radiación por explosión nuclear, o impacto de un meteorito…).
Req(14): El usuario podrá cargar antiguas simulaciones y guardarlas.
Req(15): El planeta podrá simular interacción entre sus componentes durante un número finito de vueltas actualizándose tanto él como sus componentes.
Req(16): Si el usuario decide parar la simulación, los resultados deberán guardarse.
Req(17): El modelo guardará un registro con los resultados de la simulación con distintos niveles de precisión.
Req(18): El modelo guardará la evolución de las áreas que forman un planeta como video.
Req(19): El modelo guardará las gráficas asociadas a las especies totales y vivas en una simulación.
Req(20): El modelo almacenará los datos tras cada ejecución, fusionándolo con los anteriores en caso de que ya existiesen.
Req(21): El modelo se detendrá si todas sus especies han muerto.

### 3.1.2 Requisitos no funcionales

Rnf(1): El modelo deberá hacer el menor número de comparaciones para ser lo más rápido posible.
Rnf(2): El modelo podrá ejecutarse con distintos parámetros simultáneamente.
Rnf(3): Los módulos del proyecto pueden ser intercambiados por otros en cualquier momento.
Rnf(4): Se utilizará una herramienta de control de versiones.

## *3.2 Elección de herramienta para el desarrollo*

Dados los requisitos del proyecto, hubo que tomar una decisión de diseño en que tecnología utilizar para el desarrollo del modelo. Los dos principales candidatos eran o utilizar una programación orientada a objetos, o por el contrario utilizar una programación orientada al servicio (API). El ganador, acabó siendo la programación orientada a objetos ya que se adaptaba mejor a los requisitos del modelo, puesto que cada clase formaría un elemento del sistema y cada uno tendría sus propios atributos y métodos.

En cuanto al lenguaje de programación, la elección fue clara, *Python3* (figura 3.1) ya que es un lenguaje multiplataforma, que nos ofrece muchas posibilidades al disponer de muchas librerías y existir mucho material en internet. De cara a solucionar posibles problemas, *Python2* fue descartado debido a que dejará de recibir soporte en el año 2020.


**Figura 3.1: Logo Python3 [22]**

Una vez decidida cuál sería la estructura principal de nuestro programa faltaba decidir como implementar la lógica de éste. Nuevamente había dos candidatos a utilizar: la evolución gramatical o implementar manualmente las reglas y su interpretación. Para tomar dicha decisión se investigó un programa Python llamado *PonyGE2* [18]. Tras realizar un par de pruebas, fue descartado. Esto fue debido a que añadía una lógica demasiado compleja, para lo que realmente aportaba.

Por ello finalmente se decidió utilizar python3 como lenguaje de programación, utilizando una programación orientada a objetos e implementando las reglas manualmente. Además, de utilizar git como herramienta para el control de versiones y compartir el proyecto.

## 3.3 Elaboración del diagrama de clases

Para la creación de modelos es muy importante utilizar módulos que puedan ser fácilmente intercambiados por otros. Por ello y debido a que queríamos realizar una programación orientada a objetos fue necesaria la elaboración de un diagrama de clases, teniendo una versión inicial (figura 3.2).

Dicho diagrama sufrió una modificación al crear la clase Especie (figura 3.3), con la finalidad de eliminar algunas limitaciones de nuestro modelo inicial. Gracias a dicha modificación, nuestras simulaciones podían pasar a tener infinitas especies. De la otra forma estábamos limitados a $2^{numGenes}$ (atributo del genoma de las poblaciones) especies. Antes diferenciábamos entre especies según el valor de *probEsp* de la clase población. Es decir, al comparar los genomas de dos poblaciones con el método *calcularDesigualdad()* si dicho valor era menor que probEsp, entonces considerábamos a esas dos poblaciones de la misma especie y por ende podrían reproducirse entre sí.

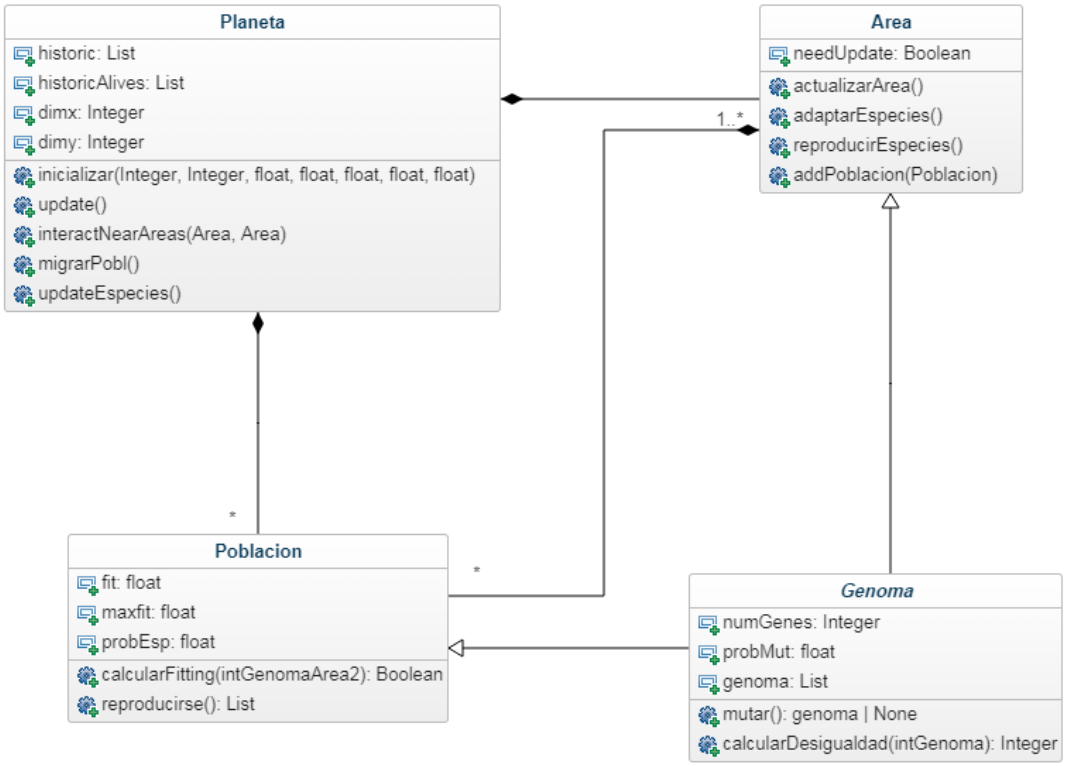


**Figura 3.2: Diagrama de clases inicial**

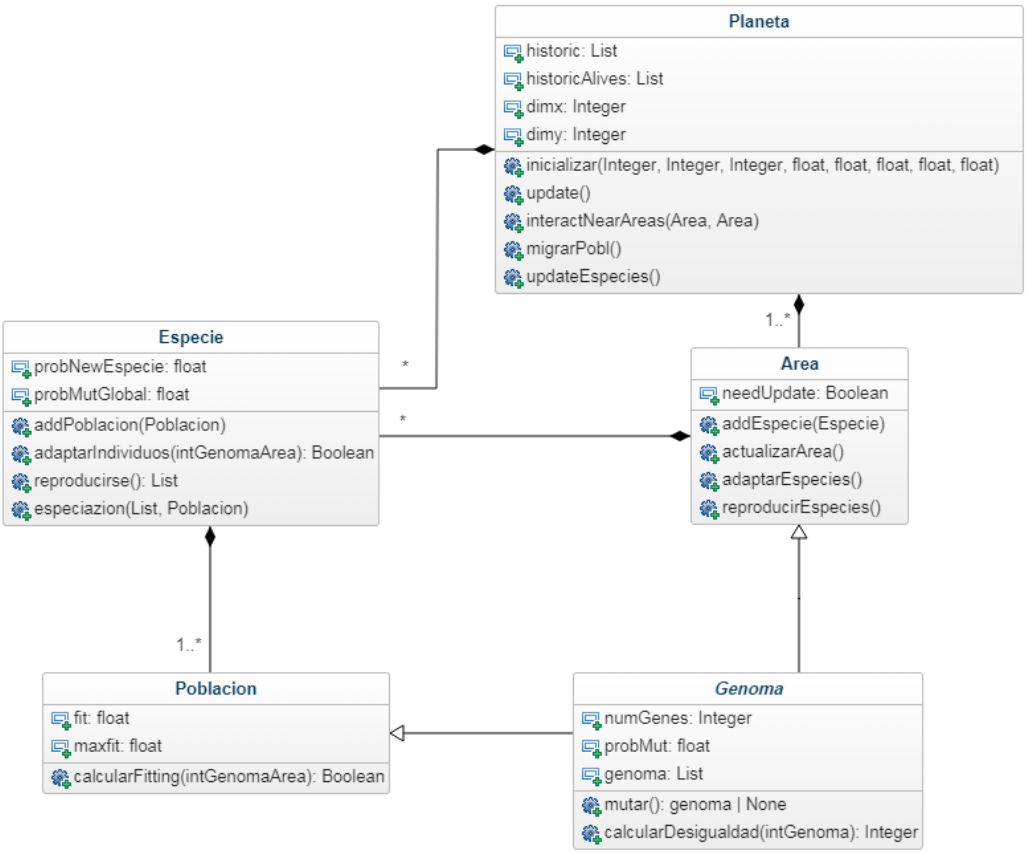


**Figura 3.3: Diagrama de clases actualizado**

# 4 Desarrollo

## *4.1 Genoma*

Esta clase es la encargada de gestionar el genoma tanto de una población como de un área. Como vimos en el diagrama de clases (figura3.3), dicha clase está compuesta por tres atributos. El fundamental es genoma, que es una lista de tamaño *numGenes* formada por enteros 0 o 1, representando a un número binario. Cuando se inicializa la clase, el genoma, puede ser recibido como argumento opcional, esto nos será útil para la reproducción que veremos más adelante. En caso de no recibirlo, la lista se inicializará con valores aleatorios, ofrecidos por la librería *random*.

Utilizamos dicho formato para el genoma debido a que operar con enteros es lo más rápido a nivel computacional, lo cual es fundamental para este modelo, además de poder realizar una conversión de binario a entero de forma eficiente, permitiéndonos dar más peso a determinados genes, como sucede en la realidad, por ejemplo en el caso de las polillas del abedul [26], donde los genes fenotípicos que determinan su color son fundamentales para su supervivencia ya que les permiten pasar desapercibido al descansar en los árboles, sin ser devoradas por sus predadores, mientras que otra característica no es tan importante para su supervivencia.

La decisión de hacer esta clase abstracta es debido a que, en el caso de las poblaciones, estas necesitan tener un genoma, que les identifique y otorgue unas características que determinaran su supervivencia. Sin embargo, en las áreas, se utiliza para representar las características de una región del planeta; es decir, un área tiene sus propias características como por ej. el porcentaje de oxígeno, nitrógeno, agua, pH… Si a cada una de estas características le asociamos un número acorde a su valor y este lo convertimos a binario obtenemos una cadena de bits que representa dicha característica en concreto. Al unir todas estas cadenas de bits obtendremos, una cadena de bits más grande, la cual contiene todas las características que definen a una determinada área en un momento determinado. Siendo esto equivalente al genoma en las poblaciones.

Además, el genoma de una población no es algo estático, puede mutar a lo largo del tiempo, como vimos en el estado del arte. De la misma manera pasa con las áreas de un planeta, un claro ejemplo de esto es la *edad de hielo*. Si lo ilustramos con un ejemplo sencillo para que se vea la similitud entre áreas y poblaciones:

Genoma inicial: 00000111 (7) / Polilla blanca / Edad de hielo

Mutación

Genoma final: 00001111 (15) / Polilla negra / Bosque de España

Las mutaciones suceden con una determinada probabilidad, por ello creamos un atributo *probMut*. Este atributo se utiliza cada vez que se activa algún evento que implica mutaciones como veremos más adelante. Para la implementación de esta funcionalidad, volvemos a apoyarnos en la librería random, la cual, por cada bit de nuestro genoma, nos devuelve un valor en el intervalo [0,1], si este es menor que la probabilidad previamente definida se aplica una *xor*, entre el gen seleccionado y 1, cambiando el valor de este y realizando el menor número de operaciones en CPU posibles.

Por lo cual, tiene sentido que tanto área como población hereden de una misma clase, ya que esto nos aporta grandes ventajas a la hora de mantener el proyecto, y evitando tener código repetido.

Finalmente nos falta hablar del método que nos calcula la desigualdad entre dos genomas, *calcularDesigualdad()*. Este método, recibe el genoma con el que queremos comparar como argumento. Realizamos una resta en valor absoluto del valor entero de ambos genomas y finalmente aplicamos una regla de tres para saber el porcentaje de diferencia entre cero y cien. Un ejemplo:

Genoma A: 0001 (1)
Genoma B: 1111 (15)
Resta en valor absoluto: |(-14)|
Regla de tres: $2^4$ ($2^{numGenes}$) ➜ 100
14 ➜ X
Es decir: 14 * 100 / 16 = 87.5

Lo cual implica que ambos genomas difieren en un 87.5%.

## *4.2 Población*

Como ya vimos anteriormente, esta clase hereda de Genoma, por ello comparte atributos y métodos con la clase anterior. Además, incluirán una característica extra, y esta es la constante de fitting, *fit*. Esta variable cuenta con un problema, y es que, si un individuo alcanza un 100% de fitting, dicha población permanecerá para siempre y su descendencia tendrá una alta posibilidad de hacer lo mismo saturando el modelo. Por ello se decidió limitar el fitting máximo de todas las poblaciones mediante la variable *maxFit*. Dicha limitación es lógica ya que existen zonas en las que no hay vida [21].

Volviendo a la variable fit, con el método *calcularFitting()*, calcula la probabilidad que tiene una población de sobrevivir en una determinada zona. Un ejemplo con fit = 0.4

Genoma población A: 0001 (1)
Genoma área B: 1111 (15)
calcularDiferencia(A, B) = 87.5
0.4 >= 87.5 ➜ False

La población A no podrá sobrevivir en el área B.

En las dos primeras versiones del modelo el valor de fit era constante y común para todas las poblaciones. Este valor era recibido como argumento de entrada del programa. En la siguiente versión el valor de este atributo se calcula utilizando una distribución gamma (figura 4.1) ofrecida por la librería s*cipy*.

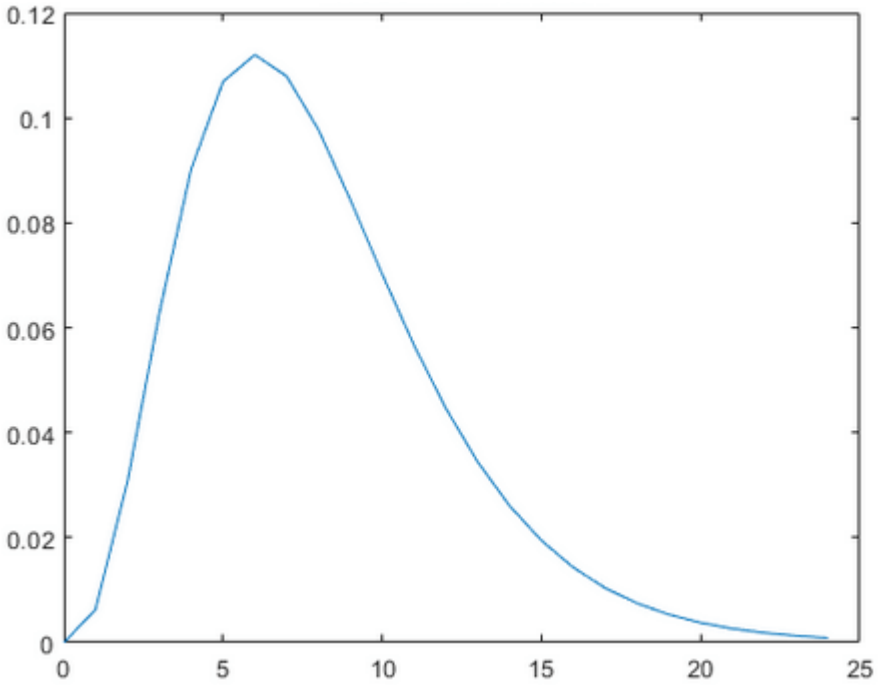


**Figura 4.1: Distribución Gamma a=5 [19]**

Donde el valor de la variable *a*, es enviado como parámetro al constructor de la clase Población. Dicho valor se obtiene como argumento de entrada del programa para las primeras poblaciones, luego se hereda el valor de fit de la población padre, en caso de tratarse de una reproducción dentro de individuos de una misma población. En caso contrario, se calcula mediante la media aritmética entre el fit de las poblaciones progenitoras.

Hemos decidido utilizar una distribución gamma ya que nos permite realizar la herencia del valor de una forma más parecida a la realidad. Es decir, tenemos un alto número de probabilidades de que el fit se herede tal cual o con una ligera variación, pero a su vez, tenemos una pequeña probabilidad de que el fitting de la nueva generación pase a ser mejor o peor, dando lugar a los fenómenos evolutivos vistos en el darwinismo [10].

## *4.3 Especie*

Esta clase surgió en una segunda versión del modelo con la finalidad de eliminar la limitación que nos impedía tener más de $2^{numGenes}$. Su integración no complicó la lógica del programa ya que simplemente añadimos un intermediario entre las poblaciones y las clases que lo importaban. Además de eliminar la limitación también nos reducía el número de operaciones que debía realizar nuestro modelo ya que como vimos en la fase de diseño, antes se calculaba si dos poblaciones eran de la misma especie en base a su genoma, ahora ya no era necesaria dicha comprobación ya que todas las poblaciones dentro de la clase especie podían reproducirse entre sí.

Después de esta aclaración sobre los motivos por los cuáles esta clase fue creada, vamos a hablar de sus atributos. En primer lugar, encontramos *probNewEspecie*, esta variable apoyada una vez más por la librería random, nos indica en el método *especiacion()*, si la nueva población producto de una reproducción, pertenece a una nueva especie, creamos esta nueva especie y la añadimos al área correspondiente y por lo tanto al planeta, también. El segundo atributo *probMutGlobal*, al igual que el anterior apoyado en la librería random, nos indica si el cambio producido en una determinada población a lo largo de una época es debido a una mutación de la población por factores externos, como, por ejemplo: la explosión de un reactor nuclear… o por el contrario se trata simplemente de una mutación debido al proceso evolutivo.

Por último, en cuanto a las funcionalidades de dicha clase es la encargada de actualizar las poblaciones que la componen, es decir; comprobar si los individuos siguen pudiendo adaptarse al área en el que se encuentra, mediante llamadas al método calcular fitting de Población. Finalmente, el método *reproducirse()*, en primer lugar, trata de reproducir a los individuos de una misma población, mediante una llamada al método *mutar()*, de genoma que ha heredado la población, si ha sucedido algún cambio se comprueba con probMutGlobal. Si se trataba de un cambio reproductivo o no y en caso afirmativo añade la nueva población a la especie, siempre y cuando esta no existiese. Mientras que, para la reproducción entre distintas poblaciones, con ayuda de la librería *itertools*, combinamos todas las poblaciones de la especie de manera eficiente y en base a sus diferencias genéticas y a un número aleatorio generado con random, comprobamos si las poblaciones pueden reproducirse entre sí. Utilizar esta técnica tiene sentido ya que cuanto más parecidas sean dos poblaciones más probabilidades tienen de reproducirse, un claro ejemplo de esto podrían ser los perros, los cuales pertenecen a la misma especie, pero hay distintas razas o poblaciones, es más probable que un pastor alemán se reproduzca con un husky siberiano que con un dachshund o perro salchicha comúnmente conocido.

## 4.4 Área

Esta clase hace referencia a una determinada región del planeta. Como vimos en apartados anteriores esta clase está compuesta de un genoma el cuál la dota de características propias y a su vez por el conjunto de especies que la habitan. De una forma análoga a como la clase Especie actualizaba las poblaciones que la componían, el área hace exactamente lo mismo a nivel de especie. Entrando en más detalles, el área tiene un método para actualizarse, *update()*, llamando en orden a las siguientes funciones: *reproducirEspecies()*, el cual se encarga de llamar al método reproducirse de cada especie, esperando a que se creen nuevas especies, en caso de que aparezcan nuevas especies se activa un flag, *needUpdate*, este nos indica que el área ha incorporado nuevas especies. Generalmente el uso de flag, no es una buena alternativa, pero en este caso es la forma más eficiente de saber si el planeta debe incorporar alguna especie nueva a su histórico de especies que veremos en el siguiente apartado.

Una vez terminado el proceso reproductivo, se intenta mutar el genoma del área, haciendo referencia a los acontecimientos que han podido hacer que dicho área cambie pudiendo significar desde la caída de un meteorito (elemento aleatorio que mutaría uno de los bits más significativos en caso de que el meteorito fuese de grandes dimensiones), a que aumente o disminuya la temperatura unos pocos grados debido a un cambio de estación. Finalmente, se llama a *adaptarEspecies()*, que al igual que *reproducirEspecies()*, llama al método adaptar individuos de cada especie.

## 4.5 Planeta

Se trata de la clase principal de nuestro modelo, en ella se almacena un registro por época, con el número total de especies vivas y que han existido. A su vez se tiene una lista con todas las especies que hay vivas en el planeta en estos momentos. Finalmente encontramos dos variables *dimx* y *dimy*, que nos indican cuál será la superficie de nuestro planeta, o cuantas áreas lo compondrán. Dichas áreas también serán almacenadas en una lista del objeto planeta. Las áreas que componen un planeta tienen una característica especial, y es que están almacenadas de manera que forman un toroide (figura4.2).

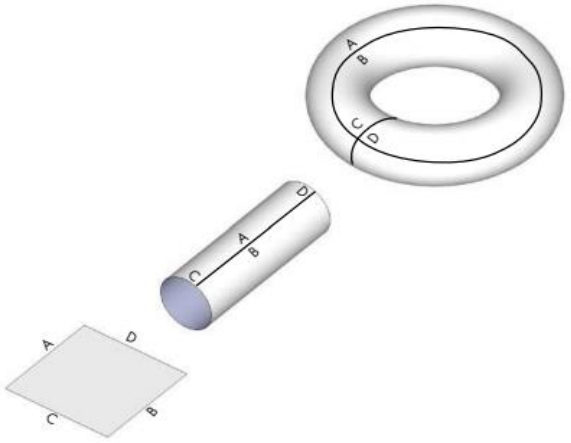


**Figura 4.2: Representación de un toroide [17]**

Esta característica nos permite tener a todas las áreas comunicadas con sus áreas contiguas. Asemejándose a la estructura real de un planeta. Además, nos permite que las especies puedan migrar a las áreas cercanas. Para esto se utiliza el método *interactNearAreas()*, por cada par de áreas colindantes calcula la diferencia que existe entre sus genomas y una vez más con la ayuda de la librería random, permite a las especies moverse entre las áreas cercanas donde más adelante tendrán que superar el proceso de fitting.

Al igual que las áreas, el planeta cuenta con un método que actualiza su contenido. A su vez, al tratarse de la clase principal también cuenta con un método que recibe como argumentos de entrada el número de poblaciones y cuantas especies quieres crear inicialmente, seguido de los parámetros necesarios para la creación de éstos, los cuáles fueron descritos en apartados anteriores.

Por último, el planeta cuenta con un método que gracias a la librería *matplotlib*, nos genera una imagen con la gráfica del número de especies totales y el número de especies vivas para cada época, que veremos en el siguiente apartado.

## *4.6 Main*

Como su propio nombre indica es el encargado de arrancar el modelo. En él se fijan cuáles serán los parámetros con los que se inicializará la ejecución, además de controlar los errores y/o excepciones que puede dar el programa, como *KeyBoardInterrupt*, la cual parará la simulación y guardará todos los datos. Ahora describiremos cuáles son las características principales del main.

Consta de dos modos de ejecución: El primero, crea un nuevo planeta acorde a los parámetros introducidos por el usuario. Mientras que el otro carga un planeta que previamente había sido creado y guardado.

Además, el main se encarga de generar para cada época una copia de los datos y generar una animación, que podemos ver en el siguiente apartado, utilizando la librería previamente mencionada, matplotlib. Dicha animación es la encargada de llamar al método *update()* de planeta.

Para el almacenamiento de los datos del planeta, se utiliza la librería *Pickle*. Además, en el main también se inicializa el registro del programa mediante la librería *logging*, la cual nos permite almacenar un fichero de forma organizada por niveles, los mensajes que se encuentran en las funciones principales del programa.

Por último, en este script también se gestiona con ayuda de la aplicación *ffmpeg*, la unión de las animaciones en el caso de estar cargando un planeta, con la finalidad de no perder la evolución de las áreas en las épocas pasadas.

# 5 Integración, pruebas y resultados

## *5.1 Integración*

Este modelo ha sido desarrollado para poder ser ejecutado en cualquier máquina con Windows o Linux que tenga instaladas las librerías mencionadas en git [27]. Durante la fase de integración, se han tenido en cuenta en qué tipo de máquina se ejecutaba, al tratarse de un único programa Python, toda la carga está concentrada en un proceso del sistema operativo, que en general tiende a ocupar un núcleo del procesador, como podemos ver en la figura 5.1, donde nos encontramos en una máquina de 8 núcleos (figura 5.2) y hay ejecutándose simultáneamente 8 simulaciones.

```
%Cpu(s): 87,4 usuario,  0,2 sist,  0,0 adecuado, 12,4 inact,  0,0 en espera,  0
KiB Mem : 16329964 total,  7254008 libre,  3624604 usado,  5451352 búfer/caché
KiB Intercambio:  7999484 total,  7999484 libre,        0 usado. 12067964 dispo

  PID USUARIO   PR  NI    VIRT    RES    SHR S  %CPU %MEM     HORA+ ORDEN
25601 juan      20   0 1239160 566532  19900 R 100,0  3,5   6030:37 python3
15607 juan      20   0 1036372 363756  19896 R  99,7  2,2   7190:12 python3
17004 juan      20   0  778612 106100  19896 R  99,7  0,6   5801:13 python3
26009 juan      20   0  783864 111364  19896 R  99,3  0,7   6027:34 python3
16677 juan      20   0  784000 111424  19896 R  95,7  0,7   5784:34 python3
15877 juan      20   0  837812 165064  19908 R  74,9  1,0   5826:19 python3
16426 juan      20   0  779252 106620  19896 R  66,3  0,7   5805:24 python3
16262 juan      20   0  871732 196532  19896 R  62,0  1,2   5792:40 python3
```

**Figura 5.1: Consumo de simulación**

```
Arquitectura:                        x86_64
modo(s) de operación de las CPUs:    32-bit, 64-bit
Orden de los bytes:                  Little Endian
CPU(s):                              8
Lista de la(s) CPU(s) en línea:      0-7
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»:              4
«Socket(s)»                          1
Modo(s) NUMA:                        1
ID de fabricante:                    GenuineIntel
Familia de CPU:                      6
Modelo:                              42
Nombre del modelo:                   Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
Revisión:                            7
CPU MHz:                             3401.000
CPU MHz máx.:                        3401,0000
CPU MHz mín.:                        1600,0000
BogoMIPS:                            6784.77
Virtualización:                      VT-x
Caché Lld:                           32K
Caché Lli:                           32K
Caché L2:                            256K
Caché L3:                            8192K
CPU(s) del nodo NUMA 0:              0-7
```

**Figura 5.2: Características CPU**

En la figura 5.1, vemos como los tres últimos procesos no llegan al 100%, esto se debe a que el sistema operativo, está ejecutando otras tareas y no puede dedicar todos los núcleos a la ejecución del programa.

Al querer realizar una simulación “larga”, hemos tenido que conectarnos a una máquina remota que permanecerá encendida, mediante el comando *ssh*, ejecutando los procesos con *nohup*, ya que de no hacerlo de esta manera al cerrar la conexión nuestros procesos morirán.

Al querer realizar pruebas más ambiciosas, con un planeta con un mayor número de áreas, el procesador se queda sin suficiente potencia siendo realmente lentas las simulaciones. Por ello, investigamos acerca de cómo ejecutar las simulaciones utilizando la potencia de nuestra GPU, que nos permite realizar cálculos matemáticos de una manera mucho más rápida. Para ello investigamos la librería llamada *Numba* [20], la cual nos permite ejecutar algunas de nuestras funciones de código Python en la GPU de nuestro ordenador. El problema de esto fue que nuestro código no estaba preparado para aceptar las características que esta librería exigía. Al requerir el uso de arrays *Numpy*, los cuáles no utilizábamos, y que nuestras funciones fuesen aún más específicas. Es decir, realizar únicamente una operación matemática…

## *5.2 Pruebas y resultados*

En este apartado, vamos a exponer las pruebas que han sido realizadas siguiendo un orden cronológico, desde que comenzó el proyecto, omitiendo las pruebas relacionadas con la comprobación del correcto funcionamiento de las operaciones lógicas de nuestro modelo, al no ser relevantes para los objetivos finales del proyecto. Seguidas de un análisis o explicación de los resultados, recomendando al lector que vuelva a revisar la figura 2.1, con el fin de recordar cuál era nuestro objetivo final.

Las primeras pruebas fueron realizadas sobre la primera versión del programa. En esta versión el número de poblaciones iniciales se calculaba de forma aleatoria por cada área creada, siendo la probabilidad un número entre [0, 3] poblaciones por área.

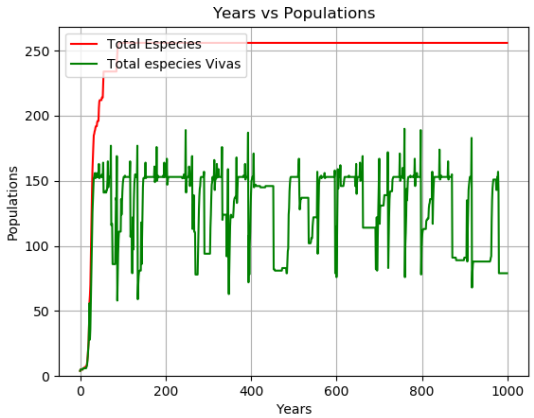


**Figura 5.3: Simulación con: numGenes=8, probMut=0.03, probEspecie=0.01, fit=0.1, nepocas=1000, dim=3x3, probMutArea=0.02**

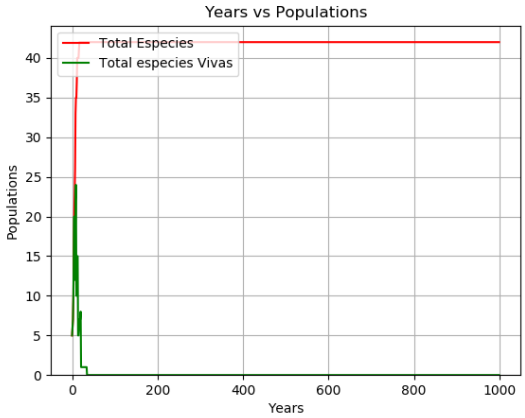


**Figura 5.4: Simulación con: numGenes=8, probMut=0.03, probEspecie=0.01, fit=0.1, nepocas=1000, dim=3x3, probMutArea=0.02**

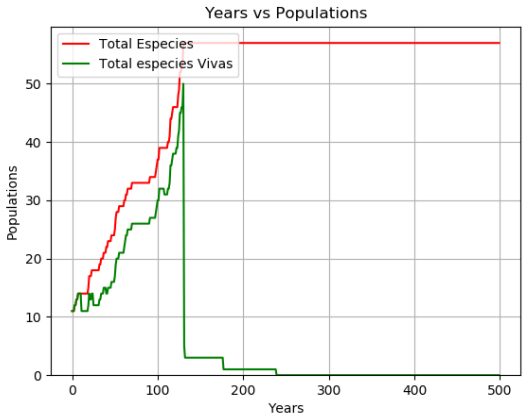


**Figura 5.5: Simulación con: numGenes=8, probMut=0.03, probEspecie=0.01, fit=0.1, nepocas=1000, dim=3x3, probMutArea=0.01**

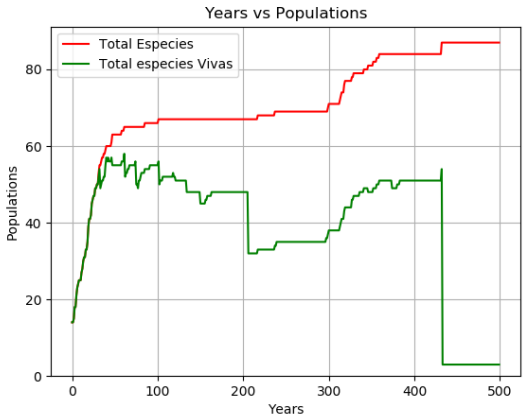


**Figura 5.6: Simulación con: numGenes=8, probMut=0.03, probEspecie=0.01, fit=0.1, nepocas=1000, dim=3x3, probMutArea=0.01**


**Figura 5.7: Simulación con: numGenes=8, probMut=0.03, probEspecie=0.01, fit=0.1, nepocas=1000, dim=3x3, probMutArea=0.005**

Como podemos observar, a medida que se disminuía el valor de la variable probabilidad de mutar área (*probMutArea*), los valores representados, fluctuaban menos, siendo más regulares, esto es lógico ya que al tener un fit tan bajo, a nada que mutase el área uno de sus bits significativos, las probabilidades de que sobreviviesen sus habitantes eran muy bajas.

En las figuras 5.5 y 5.6, podemos observar una extinción global. Las dos principales causas, son:

a) Una mutación global de todas las áreas, simulando una gran catástrofe natural como podría ser el impacto de un gran meteorito.

b) Las especies no estaban distribuidas por todo el planeta, sino que se encontraban concentradas en una o unas pocas áreas, lo cual es lógico al tener únicamente alrededor de 55 especies vivas, y desaparecer todas de golpe. Un posible escenario donde se podría dar esta situación es la figura 5.8. Se trata de un frame de la animación generada tras la ejecución del modelo. Siendo cada cuadrado el valor entero del genoma para una época determinada.

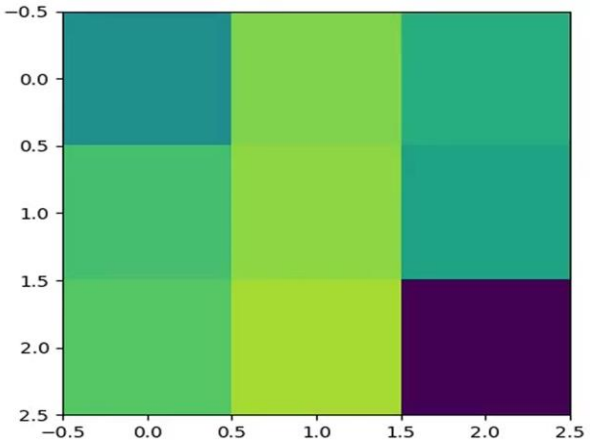


**Figura 5.8: Estado del planeta aleatorio**

Como nota, debemos recordar lo que hablamos en apartados anteriores en la primera versión del modelo únicamente podríamos tener entre 0 y 255 ($2^8$) especies sobre el planeta.

Podríamos haber sabido los motivos de estas grandes extinciones si contásemos con un log, o la animación de áreas. Esto dio lugar a la segunda batería de pruebas, esta vez, contaba con las herramientas necesarias para poder conocer en todo momento que estaba sucediendo en nuestro modelo. De esta batería, debemos destacar la siguiente gráfica (figura 5.9):

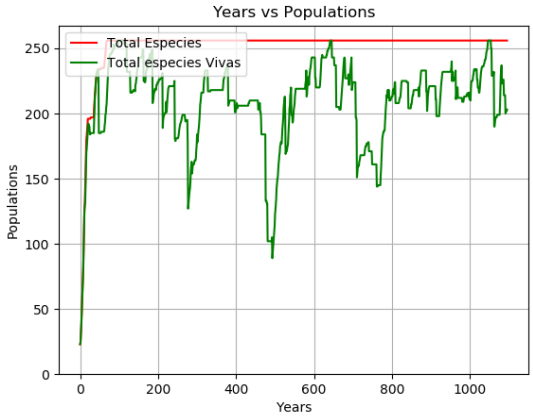


**Figura 5.9: Simulación con: numGenes=8, probMut=0.03, probEspecie=0.01, fit=0.1, nepocas=1000, dim=3x3, probMutArea=0.005**

Esta simulación, nos muestra como el sistema de mutaciones en áreas puede llegar a producir efectos devastadores en las poblaciones vivas, representando las grandes extinciones. Sin embargo, las poblaciones alcanzan el número máximo de poblaciones demasiado pronto, por lo cual no podemos apreciar el efecto incremental que ha sucedido en la historia de nuestro planeta.

En la siguiente batería de pruebas, adaptamos el valor de la constante de fit como mencionamos en el apartado anterior. Esto nos ofreció los siguientes resultados:

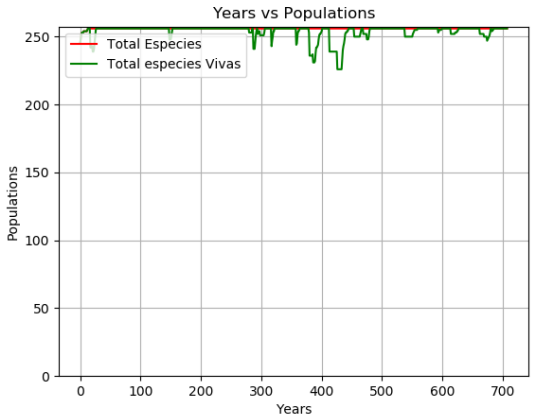


**Figura 5.10: Simulación con: numGenes=8, probMut=0.03, probEspecie=0.01, fit=0.1, nepocas=1000, dim=3x3, probMutArea=0.005**

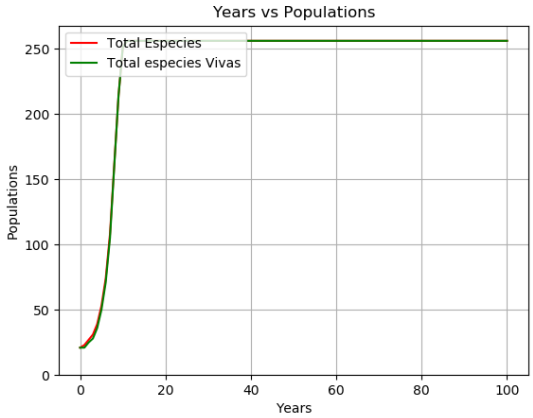


**Figura 5.11: Simulación con: numGenes=8, probMut=0.03, probEspecie=0.01, fit=0.1, nepocas=1000, dim=3x3, probMutArea=0.005**

Como podemos ver las especies eran prácticamente inmortales, ya que todas las poblaciones acababan alcanzando un 100% de fit, por ello se limitó el alcance de dicho valor. Este cambio vino acompañado con la implementación del sistema de especies, dando lugar a la versión definitiva de nuestro modelo. También, se actualizaron los valores de inicio del programa, el valor de probEspecie, fue aumentado, debido a que prácticamente era imposible que comenzase la vida en el planeta. Esto se debe a que únicamente tenemos una especie viva frente a las 5 que teníamos en versiones anteriores. Además, en estas gráficas resulta imposible ver los detalles de línea de especies vivas, por lo cual si eliminamos la línea roja (especies totales) vemos que el resultado se ve con mayor claridad por lo cual a partir de la figura 5.12 ya no la pintaremos. Dicha figura también prueba la teoría que dice “Toda especie está destinada a la extinción”, al tener unas 10000 poblaciones totales de las cuales, entorno al 10% permanecen vivas, las cuales son muestra del proceso evolutivo.

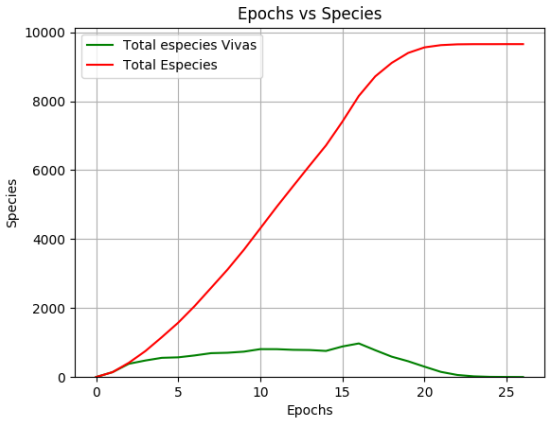


**Figura 5.12: Simulación con: numGenes=8, probMut=0.03, probEspecie=0.02, probMutGlob=0.001, minFit=0.1, maxFit=0.6, nepocas=5000, dim=10x10, probMutArea=0.005, numEspecies=1, numPobl=10**

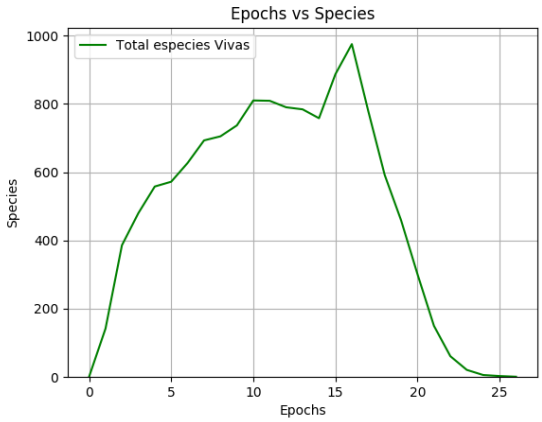


**Figura 5.13: figura 5.14 sin línea roja**

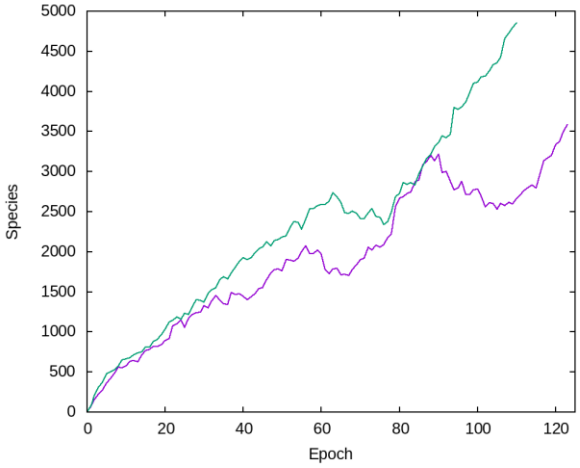


**Figura 5.14: Simulación con: numGenes=7, probMut=0.03, probEspecie=0.02, probMutGlob=0.001, minFit=0.1, maxFit=0.65, nepocas=5000, dim=10x10, probMutArea=0.01, numEspecies=1, numPobl=10**

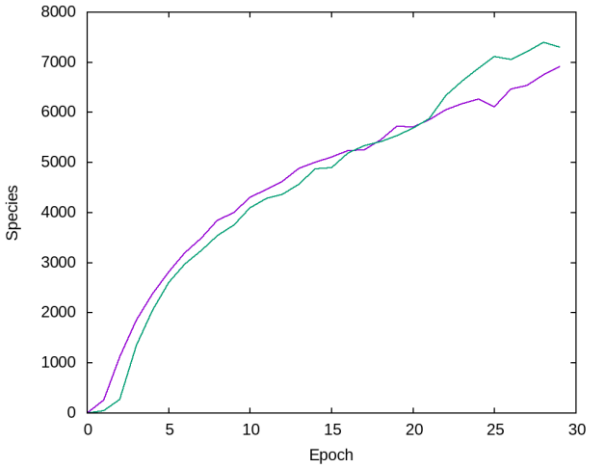


**Figura 5.15: Simulación con: numGenes=7, probMut=0.03, probEspecie=0.02, probMutGlob=0.001, minFit=0.1, maxFit=0.65, nepocas=5000, dim=20x20, probMutArea=0.01, numEspecies=1, numPobl=10**

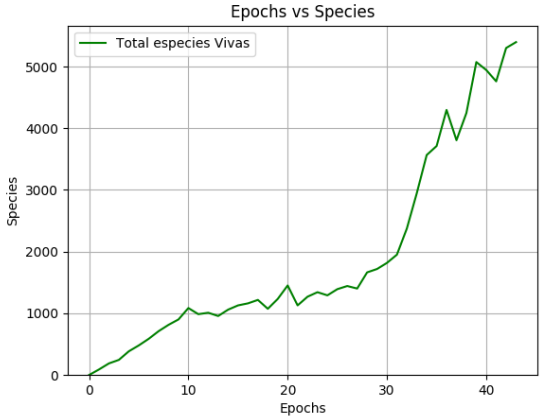


**Figura 5.16: Simulación con: numGenes=8, probMut=0.03, probEspecie=0.02, probMutGlob=0.001, minFit=0.1, maxFit=0.65, nepocas=5000, dim=4x5, probMutArea=0.005, numEspecies=1, numPobl=10**

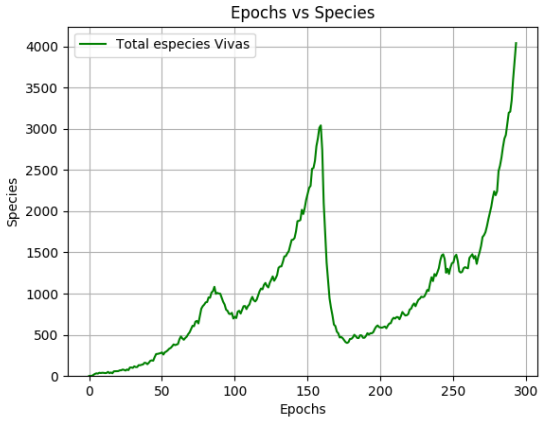


**Figura 5.17: Simulación con: numGenes=6, probMut=0.03, probEspecie=0.02, probMutGlob=0.001, minFit=0.1, maxFit=0.6, nepocas=5000, dim=3x3, probMutArea=0.005, numEspecies=1, numPobl=10**

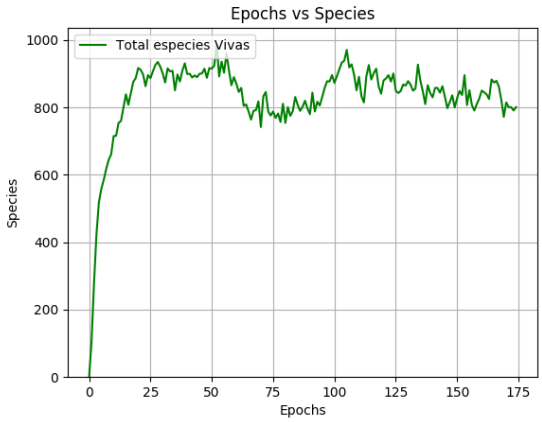


**Figura 5.18: Simulación con: numGenes=6, probMut=0.03, probEspecie=0.02, probMutGlob=0.001, minFit=0.1, maxFit=0.65, nepocas=5000, dim=8x8, probMutArea=0.005, numEspecies=1, numPobl=10**

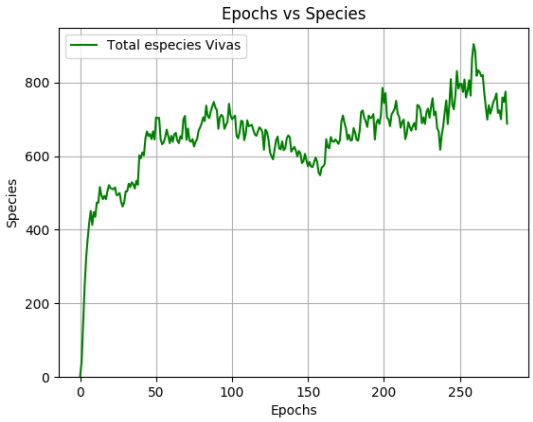


**Figura 5.19: Simulación con: numGenes=6, probMut=0.03, probEspecie=0.02, probMutGlob=0.001, minFit=0.1, maxFit=0.65, nepocas=5000, dim=6x6, probMutArea=0.005, numEspecies=1, numPobl=10**

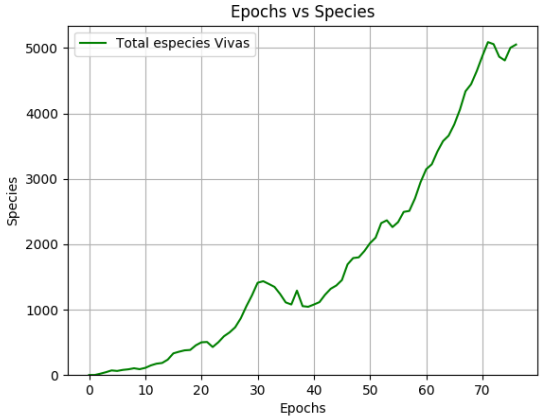


**Figura 5.20: Simulación con: numGenes=7, probMut=0.03, probEspecie=0.01, probMutGlob=0.001, minFit=0.1, maxFit=0.65, nepocas=5000, dim=7x7, probMutArea=0.005, numEspecies=1, numPobl=10**

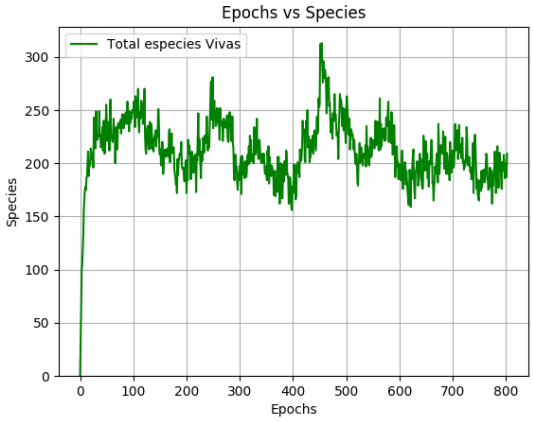


**Figura 5.21: Simulación con: numGenes=6, probMut=0.03, probEspecie=0.007, probMutGlob=0.001, minFit=0.1, maxFit=0.65, nepocas=5000, dim=5x5, probMutArea=0.01, numEspecies=1, numPobl=10**

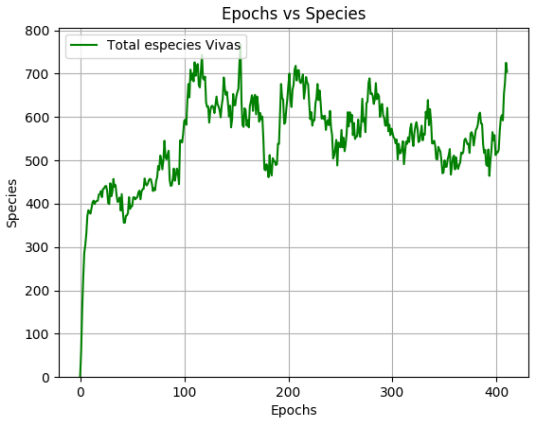


**Figura 5.22: Simulación con: numGenes=6, probMut=0.03, probEspecie=0.02, probMutGlob=0.001, minFit=0.1, maxFit=0.65, nepocas=5000, dim=10x10, probMutArea=0.01, numEspecies=1, numPobl=10**

Las evidencias con respecto a los distintos parámetros de entrada que nos aportan estos resultados son:

- Al aumentar el tamaño del tablero los tiempos se reducen a la mitad en el caso de duplicar el tamaño, y a su vez el número de especies tiende a multiplicarse (figuras 5.14 y 5.15).

- Si disminuimos el tamaño del genoma los tiempos se aceleran, pero a cambio perdemos especies ya que al calcular la diferencia entre el área y la población hay menos margen de error. Por ejemplo, con un fit del 25%, si tenemos 4 genes solamente una población podrá adaptarse mientras que, si tenemos 8, habrá dos poblaciones, haciendo que dicha especie sea más difícil de extinguir.

- Cuando incrementamos la probabilidad de mutación del área, las gráficas tienden a tener un mayor número de extinciones consecutivas (figuras 5.21 y 5.19).

- De manera opuesta al punto anterior, sucede cuando aumentamos la probabilidad de que surjan nuevas especies (figuras 5.22 y 5.21).

- Al establecer unos valores altos de numPobl, o numEspecies, es más probable que nuestro modelo no sufra una extinción masiva en las primeras épocas.

- A mayor número de maxFit, menor será el número de extinciones.

Como resumen final podemos decir que un modelo construido teniendo en cuenta una serie de reglas sencillas que reflejan conceptos evolutivos básicos (mutación, adaptación, fitting...) puede reproducir los patrones que se observan en el registro fósil. Ahora bien, existen tantos factores aleatorios que influyen en la macroevolución, que es prácticamente imposible reproducir de manera exacta cuál ha sido la historia evolutiva de la Tierra. Ni aun teniendo una máquina que ejecutase millones de épocas por milisegundo, y pudiésemos descartar todas las gráficas que no fuesen iguales al registro fósil, sería imposible saber cuál de las infinitas simulaciones aludiría a qué pasaría en la próxima época.

También, debemos resaltar la figura 5.17, siendo la que más se asemeja al registro fósil, e ilustrando cómo las extinciones masivas pueden ser sistémicas, no hace falta un factor externo para que se produzcan. Tras alcanzar un punto de “suficientes especies”, teniendo en cuenta las características del modelo, pasamos de tener entrono a 3000 especies vivas a unas 500 (~84% de extinción, por definición, una extinción masiva). Al no haberse producido ningún cambio radical, en las características de las áreas que indique un cataclismo (figura 5.23).

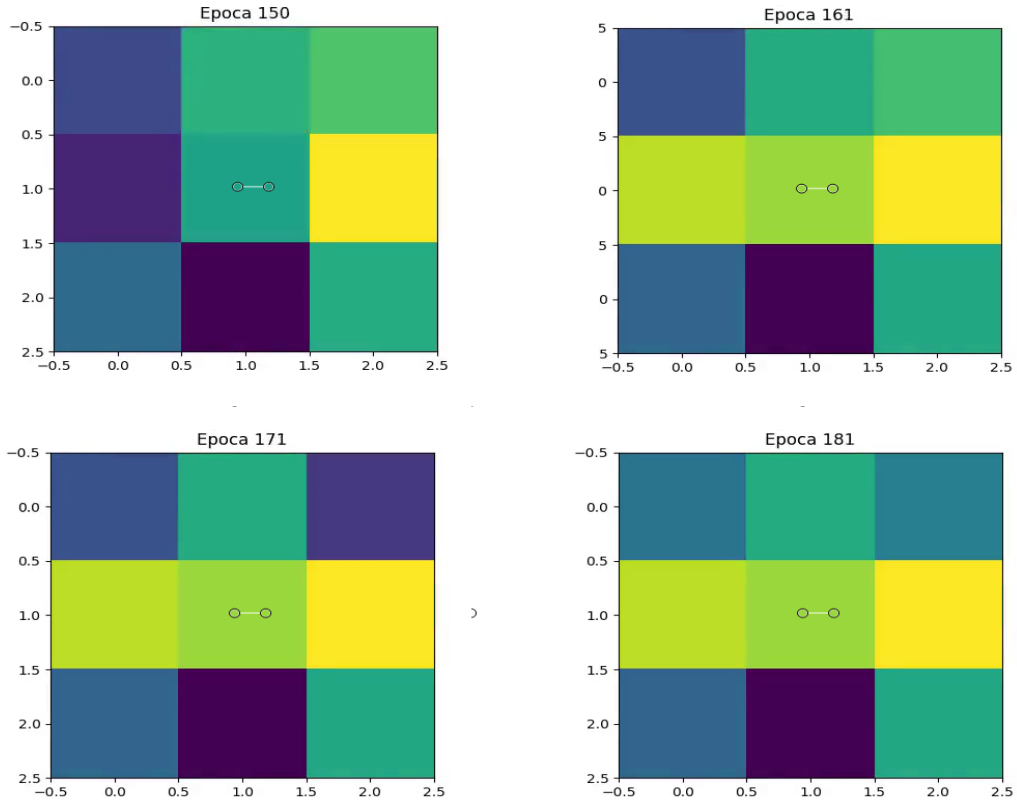


**Figura 5.23: Evolución áreas figura 5.17, épocas de la extinción masiva.**

# 6 Conclusiones y trabajo futuro

## *6.1 Conclusiones*

Este trabajo tenía como propósito la implementación de un modelo macroevolutivo con el cuál estudiar el comportamiento de las especies en un planeta.

Para lograr dicho objetivo, se ha estudiado el estado del arte, repasando tanto el contexto biológico como distintas aproximaciones teóricas al problema, y se ha diseñado e implementado un modelo capaz de simular las dinámicas aprendidas. A medida que ha ido avanzando el proyecto, dicha implementación ha sufrido cambios con el fin de optimizarlo en cuanto a rendimiento y producción de resultados.

En base a los resultados obtenidos, podemos concluir que hemos logrado satisfacer los objetivos que nos planteábamos al principio del proyecto. Por ello el modelo sugiere que, al menos de forma teórica, las grandes extinciones podrían ser una componente propia de la dinámica del sistema.

Esto no ha sido lo más importante, puesto que lo más importante han sido todos los nuevos conocimientos que he adquirido acerca de biología e informática.

Es evidente que los cambios bruscos en el entorno afectan de forma directa y letal a las especies que habitan un planeta, pero que éste tiene capacidad plástica para recuperarse. Por ello, debemos respetar el medio ambiente para no ser los responsables de una sexta gran extinción masiva.

## *6.2 Trabajo futuro*

En base al modelo actual, se sugieren varias mejoras con el fin de crear un modelo aún más preciso y concreto.

- Implementar una red trófica, siguiendo las dinámicas de Lotka-Volterra.

- En base a dichas dinámicas tener en cuenta la epigenética, ya que desempeña un rol importante en la evolución.

- Adaptar el código para su compatibilidad con Numba [20], mejorando la velocidad del modelo.

- Cambiar el tipo de dato del genoma a un binario puro, y operar utilizando puertas lógicas para aumentar aún más la eficiencia del modelo.

- Realizar simulaciones con varios planetas, para tratar de contestar la pregunta: ¿ Existe la vida en otros planetas? ¿ Con que probabilidad?

# Referencias

Todos los enlaces están disponibles a día 11 de junio de 2019.


[1] Muy Interesante, https://www.muyinteresante.es/ciencia/fotos/las-mayores-extinciones-de-la-historia/primera-gran-extincion
[2] Wikipedia, Fenerozoico, https://es.wikipedia.org/wiki/E%C3%B3n_Fanerozoico
[3] Wikipedia, geología histórica, https://es.wikipedia.org/wiki/Geolog%C3%ADa_hist%C3%B3rica
[4] Thomas M. Smith y Robert Leo Smith, Ecología, pp. 194-299, 2007
[5] Wikipedia, fitting, https://en.wikipedia.org/wiki/Ecological_fitting
[6] Wikipedia, especiación, https://en.wikipedia.org/wiki/Ring_species
[7] J.A. Coyne y H.A. Orr, SPECIATION, 2004
[8] D.J. Futuyma, EVOLUCIÓN, 1 abril 2006
[9] S. Freeman y J.C. Herron, ANÁLISIS EVOLUTIVO. 22 agosto 2013
[10] Espaciociencia, darwinismo y neodarwinismo, https://espaciociencia.com/darwinismo-neodarwinismo
[11] Wikipedia, Lotka-Volterra, https://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equations
[12] Wikipedia, Generalized Lotka-Volterra, https://en.wikipedia.org/wiki/Generalized_Lotka%E2%80%93Volterra_equation
[13] sciPy, implementación Lotka Volterra, https://scipy-cookbook.readthedocs.io/items/LoktaVolterraTutorial.html
[14] Manuel Alfonseca y José Soler Gil, Evolving a Predator–Prey Ecosystem of Mathematical Epressions with Grammatical Evolution, 25 enero 2014
[15] Li Chen1 y Tai-Sheng Wang, Modeling Strength of High-Performance Concrete Using an Improved Grammatical Evolution Combined with Macrogenetic Algorithm, junio 2010
[16] Shao-Meng Qin, Yong Chen y Pan Zhang, Network growth approach to macroevolution, julio 2007
[17] Nostrach, el juego de la vida, https://nostarch.com/download/samples/PythonPlayground_sampleCh3.pdf
[18] GitHub, gramática evolucional Python, https://github.com/PonyGE/PonyGE2/wiki
[19] Datacamp, distribuciones en Python, https://www.datacamp.com/community/tutorials/probability-distributions-python
[20] Numba, https://numba.pydata.org/numba-doc/dev/user/5minguide.html
[21] ISME, Jacqueline Goordial, Nearing the cold-arid limits of microbial life in permafrost of an upper dry valley, Antarctica, 19 enero 2016
[22] Python, https://www.python.org/
[23] Metode, Tamaño del genoma en algunos seres vivos, https://metode.es/revistas-metode/monograficos/el-tamano-del-genoma-y-la-complejidad-de-los-seres-vivos.html
[24] La Vanguardia, Especies extremófilas, https://www.lavanguardia.com/natural/20160516/401832534098/animales-extremo-naturaleza.html
[25] Wikipedia, el juego de la vida, https://es.wikipedia.org/wiki/Juego_de_la_vida
[26] Charles Darwin, El origen de las especies, 24 noviembre 1859
[27] Git del proyecto, https://github.com/rlatorre-uam/juan_macroevolucion

# Glosario

| | |
|---|---|
| API | Application Programming Interface |
| UML | Unified Modeling Language |
| Fitting | Capacidad de adaptación a un determinado hábitat |
| GE | Grammatical Evolution |
| Dataset | Conjunto de datos |
| GEGA | Grammatical Evolution Genetic Algorithm |
| CPU | Central Process Unit |
| GPU | Graphics Processing Unit |
| KeyBoardInterrupt | Detener la ejecución de un programa, presionando cntrl+C |
| SSH | Secure Shell |
| Frame | Fotograma |
| ADN | Ácido desoxirribonucleico |